UNITED STATES PATENT APPLICATION

for

VIRTUAL PCI DEVICE APPARATUS AND METHOD

Inventors:

Brian K. Langendorf
Varghese George

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025-1026

Attorney Docket No.: 42390P10570
Intel Disclosure No.: 15992

09895395.062901

"Express Mail" mailing label number: EL48575410765

Date of Deposit: 6-29-01

I herby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Clara Wallin

(Typed or Printed Name of Person mailing paper or fee)

Clara Wallin

(Signature of Person mailing paper or fee)

6-29-01

(Date Signed)

VIRTUAL PCI DEVICE APPARATUS AND METHOD

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The invention pertains generally to computers. In particular, it pertains to data transfer busses.

Description of the Related Art

[0002] The PCI Local Bus Specification Rev. 2.2 and PCI to PCI Bridge Architecture Specification Rev. 1.1, both published by the PCI Special Interest Group, proscribe a PCI (Peripheral Component Interconnect) bus protocol for integrating network controllers, mass storage controllers, display controllers, multimedia device, communication device, and other devices into a system. PCI bus protocol, which includes specifications for electrical characteristics as well as the manner in which software interacts with PCI devices, provides for the integration of peripheral devices in a manner that is generally independent from the particular protocol employed by other system components such as a host bus, processor, or memory.

106250.56356860

[0003] PCI protocol provides "hooks" to enable computer programs to directly access and configure PCI devices. These hooks include configuration address space for accessing 256 8-bit configuration registers associated with each PCI device to allow computer programs to optionally detect each PCI device installed in a system, identify the vendor and device type, determine each PCI device's system resource requirements, relocate each device within the system address space, interrupt binding, install, configure, boot without user intervention and for system address map construction. Configuration registers include a predefined header region and device dependent region; however, only the necessary and relevant registers in each region need be implemented. PCI-to-PCI bridges are PCI devices with configuration registers, which are accessible to a computer program to assign a range bus number to the PCI busses behind the PCI-to-PCI bridge.

[0004] Configuration space is accessed through configuration cycles initiated by a computer program executing on a host processor. PCI protocol anticipates that the routing of configuration cycles be accomplished through either one of two distinct mechanisms in which a host-to-PCI bridge translates a software command (in the form of a sequence of processor initiated host bus accesses to I/O space) into a single configuration cycle on the targeted PCI bus involving the assertion of a particular PCI bus signal that is received by the targeted PCI device to indicate to that device that it is the target for the current configuration cycle.

[0005] Detection and initialization of PCI devices in a system can be accomplished by device independent program by utilizing configuration space. A program may poll the configuration space allocated to each slot on PCI bus 0 to detect the presence of PCI devices and PCI-to-PCI bridges that reside on bus 0. The program may assign each detected PCI-to-PCI bridge a unique bus number by writing to particular configuration registers and then poll each slot on each assigned bus to detect the presence of PCI devices and PCI-to-PCI bridges. This process may be continued until each slot on each detected PCI bus is polled and all PCI devices are detected. For each PCI device found installed in the system, the device's configuration registers may be read to determine its resource requirements. A system address map may be built to eliminate conflicts among the system and various PCI devices and system resources may be assigned to each PCI device by writing

10523095.095601

to the appropriate configuration registers in each PCI device. Additionally, a self-test may be invoked on those devices that support self-test and any other initialization, installation, and configuration may be accomplished for each PCI device with or without user intervention.

[0006] Disadvantages imposed by strict adherence to PCI protocol include limits on data transfer speed, data path width, latency, and bandwidth, which set an upper boundary for performance of PCI devices. Performance may also be restricted by delays inherent to host-to-PCI bridges as well as the bandwidth constraints inherent to multiple PCI devices residing on a shared PCI bus. The present trend is for increasingly higher performance processors, memory, and host busses in which efficiently coupled devices can achieve performance advantages such as lower latency, higher throughput, and increased overall system performance than can be achieved through coupling to an actual PCI bus.

106290.5959860

Sub A1

~~**[0007]** Plug-and-Play™ resource allocation programs, as required by PCI bridge specifications, typically expect the address space allocated to a particular PCI bus to include the address space allocated to any PCI bus behind that particular PCI bus. Accordingly, full compliance with PCI protocol increases the difficulty of locating PCI devices on the host processor side of a host-to-PCI bridge, where the PCI devices - possibly for compatibility reasons - require an address space that may be a subset of the address space allocated to a physical PCI bus.~~

BRIEF DESCRIPTION OF DRAWINGS

Figs. 1a, 1b show a system configuration in accordance with the present invention.

Fig. 2 shows a more detailed system configuration.

Fig. 3 shows a system with a primary virtual bridge.

Fig. 4 shows a system with a secondary virtual bridge and a primary virtual bridge.

Fig. 5, 5a, 6b show flow charts of method embodiments.

DETAILED DESCRIPTION

[0008] Various embodiments of the provide for the integration of devices, such as network controllers, mass storage controllers, display controllers, multimedia devices, communication devices, and other devices, into a host bus processor as well as the efficient coupling of a device to a processor host bus. Select aspects of PCI protocol may be adhered to, enabling software support for these devices that is generally available only to PCI compliant devices. The present invention may provide, in some systems, at least one of a number of advantages over methods of prior art including: increased system performance, increased device performance, simpler initialization and configuration of all devices in a system, increased robustness in the system resource allocation process, lower overall cost and decreased physical board/chip space as well as enabling the allocation of a subset of the address space assigned to a physical PCI bus to a device residing on a host bus. Additionally, the present invention may enable integration of devices, which appear to computer programs as having many of the characteristics of a PCI compliant device, into a host processor, resulting in a number of advantages such as lower overall system cost, less space, reduction in processor-chip pin-count and increased bandwidth on PCI busses, increased system performance or increased device performance.

Sub A2

[0009] Figs. 1a, 1b, 2, 3 and 4 show block diagrams of systems 100, 200, 300 and 400 for explaining various embodiments of the present invention. Processor 130 may represent any one processor coupled to host bus 120. Alternatively processor 130 may represent two or more processors coupled to host bus 120.

[0010] Systems 100, 200, 300, 400 may include a host bus device 110, 210, 410 such as a network controller, mass storage controller, display controller, multimedia device, communication device, or other device. Host bus device 110, 210, 410 may be coupled to host bus 120 through an interface 112 in a manner that allows host bus 120 to be monitored and allows processor initiated host bus cycles, which are targeted to non-existent virtual PCI device 160, to be intercepted by host bus device 110.

10520.05953680

[0011] Host bus device 110, 210, 410 may include a monitor circuit 114 that is coupled to host bus 120 for tracking host bus cycles. Monitor circuit 114 may capture select information during each host bus cycle and identifies select host bus cycles that are to be snooped or intercepted. Read cycles may be intercepted by driving select host bus data signals to transfer data to host bus 120 and then completing the cycle. A cycle, initiated by processor 130, is completed by informing the active processor 130 when to terminate the cycle pursuant to the particular protocol employed by the host bus 120. Write cycles may be intercepted by latching the value of select host bus 120 data signals on host bus 120 write cycles and then completing the cycle. Host bus 120 cycles may be snooped by reading and storing select host bus cycle information in a storage 115 while the cycle is typically completed by another device coupled to the host bus 120.

[0012] Host bus device 110, 210, 410 may include storage 111 that is coupled to host bus 120 to allow the contents of storage 111 to be accessed through the host bus 120. Various specific embodiments of storage 111 may include registers 216, 218, 417 that reside in system configuration space or random access memory (RAM), registers, or data ports that reside in system I/O or memory address space.

[0013] PCI protocol provides for a certain number of addressable slots on each PCI bus in which a PCI device may reside. A PCI-to-PCI bridge is a PCI device that provides a transfer path between two PCI busses. A computer program typically assigns each PCI bus directly behind a detected PCI-to-PCI bridge a unique bus number. The PCI bus number and slot number combination provide a unique identifier that may be used by computer programs to select any particular PCI device installed in a system through configuration space. PCI addressing and routing protocol generally anticipates a physical hierarchal bus structure in which host bus cycles, targeted to a particular PCI device are physically routed through a host-to-PCI bridge and possibly through one or more PCI-to-PCI bridges to generate a cycle on the particular PCI bus in which the targeted PCI device resides. Each PCI bus is defined to be behind any PCI bus in the physical/virtual data transfer path between that PCI bus and host bus.

10092905.095960

[0014] For purposes of this specification a primary PCI bus 150 may be bus number 0 and be the bus directly behind the host-to-PCI bridge, but it may also be any PCI bus that is behind bus number 0 in which virtual PCI device 150 appears to be behind and it is consistent with the methods of the present inventions for there to be more then one primary PCI bus in a particular system. Additionally the primary PCI bus 160 may be an actual PCI bus or it may be virtual. The terms virtual and logical, for purposes of this specification, refer to the perspective of a computer program running on any one or more processor 130 where the program has an impression of a physical device or structure that may not be reflective of an actual physical device or structure. Virtual PCI device 160 logically resides on a PCI bus 151, which is either an extension of primary PCI bus 150 or one of its subordinates (i.e. busses that reside behind primary PCI bus 150).

[0015] A host-to-PCI bridge 140, 240, 340 may facilitate the translation and routing of a select host bus 120 cycles to a primary PCI bus 150 and its subordinate busses. Host-to-PCI bridge 140, 240, 340 may include, a host bus interface 141, a PCI bus interface 142, storage 145, storage 149 and control circuit 148 to track host bus 120 cycles to determine, for each host bus cycle, whether to route the host bus cycle to an actual primary PCI bus 150. To make this determination, control circuit 148 consults storage 145, which identifies address space allocated to primary PCI bus 150 and its subordinates, and storage 149, which identifies address space allocated to virtual PCI device 160 or possibly other virtual devices. Storage 145 and 149 may hold information in the form of bus numbers, memory address ranges, I/O address ranges or other similar information. The information in storage 149, that indicates the address ranges allocated to virtual PCI device 160 (or in more complex configurations to all or a group of virtual devices or busses), may generally be a subset of the address space in storage 145, which is allocated to primary PCI bus 150 and its subordinates. Host bus cycles which are targeted to virtual PCI device 160 are allowed to be intercepted by host bus device 110, 210, 310.

[0016] In a system where host bus 120 utilizes the protocol of a Pentium ™ 3 or 4 processor, host-to-PCI bridge 140, 240, 340 may be the responder for all host bus cycles insuring that all host bus cycles are completed according to host bus protocol, and host bus device 110 may claim select host bus cycles which it intercepts.

[0017] The information required to identify host bus cycles targeted to virtual PCI device 160 and the manner in which this information is obtained, may depend on the bus structure of the particular system. The unique bus-device number combination of virtual PCI device 160 may be hardwired or preprogrammed into both the host bus device 110 and host-to-PCI bridge in a system 100 in which virtual PCI device 160 resides logically on primary PCI bus 150.

[0018] It is consistent with the methods of the present invention for a system 100, 200, 300, 400 to have two, three or more host bus devices 110, 210, 410 or to have two, three or more processors 130 with each having one or more host bus devices 110, 210, 410 integrated into it. The methods of the present invention may be applied to a system 100 comprised of a plurality of distinct host bus devices 110, which are each to be associated with one of a plurality of distinct virtual PCI devices 160. Host bus 120 cycles targeted to any one of the plurality of virtual PCI devices 160 are not forwarded to primary PCI bus 150 by host-to-PCI bridge 140; rather, one of the host bus devices 110 is allowed to intercept the cycle. More than one host bus device 110 may be associated with a particular virtual PCI device 160, such that the two or more host bus devices 110 utilize the system resources allocated to the particular virtual PCI device. Conversely, a single host bus device 110 may be associated with more than one virtual PCI device 160. In addition, a host bus device 110 may support multiple functions in accordance with PCI protocol. A scheme for preventing multiple host bus devices 110 from intercepting cycles to the same virtual PCI device 160, may be accomplished by hardwiring or preprogramming each host bus device 110 with a

09859365.062501

unique bus number-device number combination for identifying its associated virtual PCI device 160.

Sub A3

[0019] Fig. 1a shows host bus device 110 coupled to host bus through interface 112, which is distinct from the host bus interface to processor 130. Fig. 1b shows an alternate configuration consistent with the methods of the present invention, where processor 130 and host bus device 110 are coupled to host bus 120 through an internal bus 113 and a shared host bus interface 112. The system of Fig. 1b can result from integrating host bus device 110 and processor 130 into a single circuit package.

09693558.062901

[0020] Fig 2 shows a system 200 for explaining an embodiment of the present invention pursuant to the teachings for system 100 shown in Fig. 1. System 200 includes a virtual PCI device 160, which appears from the perspective of a computer program running on one or more processors 130, to include 256 8-bit configuration registers 268 that are accessible through configuration space. Only the necessary and relevant configuration registers 268 appear to be implemented. In accordance with PCI protocol, a computer program may initiate accesses to virtual configuration registers 268 to accomplish one or more functions that include detecting the presence of virtual PCI device 160, identifying the vendor and device type determining virtual device's 160 system resource requirements, providing for full device relocation, interrupt binding, installation, configuration, booting without user intervention, and including virtual PCI device in the system map construction.

[0021] Host bus device 210 may include host bus storage 111, which in this embodiment, is comprised of PCI compliant configuration registers 218. Only the necessary and relevant registers 218 are implemented. Monitor circuit 114 tracks host bus cycles and identifies host bus cycles targeted to virtual configuration registers 268, which are intercepted and redirected to access the host bus device configuration registers 218. As a result, a computer program executing on processor 130 can access host bus device configuration registers 218 by initiating a host bus cycle targeted for the configuration registers 268 of virtual PCI device 160. In this manner, host bus device 210 participates in an initialization and configuration procedure for assigning system resources that is generally available only to PCI devices.

09868395.092601

[0022] Configuration cycles are generated in system 200 through either one of two mapping mechanisms provided by PCI protocol. Mechanism one is an indexing scheme in which two fixed locations in processor I/O space are reserved for a configuration-address register 243 and a configuration-data register 244, which are typically incorporated into a host-to-PCI bridge 240. Configuration-address register 243 enables or disables configuration space and is written by a computer program to identify a particular PCI device and specific configuration register by specifying the bus number, device number, function number, and register number for which a subsequent configuration cycle is intended. Subsequent DWORD read and write host bus cycles targeted to configuration-data register 244 are translated and routed, typically by host-to-PCI bridge 240, into PCI compliant configuration cycles; however, host bus cycles targeted to the configuration registers 268 of virtual PCI device 160 are intercepted by host bus device 210 rather then be routed by host-to-PCI bridge 240 as anticipated by PCI protocol.

[0023] A system 200 that supports mechanism one, may provide a mirror register 216 that is included in storage 115, which holds select information obtained through snooping. Host bus 120 write cycles targeted to configuration-address register 243 may be snooped by host bus device 210 and the data transferred in the snooped cycle may be stored in mirror register 216 which then reflects the contents of configuration-address register 243. Monitor circuit 114 may consult mirror register 216 to identify subsequent DWORD host bus 120 cycles targeted to configuration-data register 244 that are intended to access virtual configuration registers 268. These identified cycles may be intercepted by host bus device 210 and redirected to access corresponding configuration registers 218 in host bus device 210.

[0024] Alternately, system 200 may support mechanism two, where a configuration-space-enable register (not shown) and a forward register (not shown), which typically reside in host-to-PCI bridge 240, are written to by a computer program to specify 4k byte of configuration space to be mapped into a fixed location in processor I/O address space. A system 200, designed to supports mechanism two may provide a mirror register 216 to store snooped host bus write cycles targeted to configuration-space-enable register (not shown) and forward register (not shown). Mirror register 216 may be consulted by monitor circuit 114 to identify host bus cycles that are targeted to virtual device's configuration registers 268. Identified cycles may be intercepted by host bus device 210 and redirected to access corresponding configuration registers 218 in host bus device 210.

10052093.063650

[0025] Pursuant to PCI protocol, configuration registers may be utilized by a computer program to allocate system resources that include interrupts, processor memory address space, processor I/O address space, and ROM (read only memory) address space which is a range of processor memory address space reserved for ROMs. A computer program may initiate accesses to non-existent virtual configuration registers 268 to determine virtual PCI device's 160 system requirements and allocate resources to virtual PCI device 160 by writing to select configuration registers 268. A computer program may also access virtual device's configuration registers 268 for handling catastrophic errors as well as executing and obtaining status of a built in self-test (BIST).

[0026] Optionally, host bus device configuration registers 218 may be implemented to indicate a request for specific system resources such as one or more ranges of memory space or I/O space to be assigned to virtual PCI device 260. A portion of internal storage 111 may be mapped by a computer program into this address space for access through the host bus 120. After a computer program assigns address space to virtual PCI device 160, monitor circuit 114, in this optional embodiment, may consult the appropriate configuration registers 218 to identify whether a host bus cycle is targeted to memory or I/O space allocated to virtual PCI device 160. These identified cycles may be intercepted by host bus device 210 and redirected to access host bus storage 111.

[0027] Both the host bus device 210 and host-to-PCI bridge 240 may be aware of the bus number and device number of virtual PCI device 160. This information may be hardwired, preprogrammed, or provided by a program during system initialization and is used to identify host bus cycles targeted to virtual PCI device 160 configuration space. Host-to-PCI bridge 240 may learn the address space allocated to virtual PCI device 160 by snooping select host bus write cycles targeted to the particular virtual configuration registers 268 which specify the address space allocated to virtual PCI device 160.

[0028] Fig. 3 shows a system 300, for explaining an embodiment of the present invention, pursuant to the teachings for system 100 and system 200. System 300 implements mechanism one for accessing configuration space. System 300 may be comprised of a primary virtual PCI-to-PCI (P-P) bridge 370 that may appear, to a computer program running on a processor 130, as residing on primary PCI bus 150 and may appear as a bridge to primary virtual PCI bus 357. Virtual PCI device 360 may appear, to a computer program running on processor 130 as residing on primary virtual bus 357.

[0029] Both the host bus device 210 and host-to-PCI bridge 340, in this embodiment, are aware of the bus number and device number in which primary virtual P-P bridge 370 resides and host bus device 210 is aware of the device number of virtual PCI device 360. This information may be provided by an initialization program or be hardwired or preprogrammed.

[0030] Host-to-PCI bridge 340 may include storage 149, which in this embodiment is written by a computer program, such as a Plug-and-Play ™ resource allocation program, to assign address space to primary PCI bus 150, which encompasses all address space assigned to virtual device 160. Information stored in storage 149 may include the bus numbers of virtual busses (not shown) that may reside behind primary PCI bus 357: bus numbers are commonly sufficient information to determine the configuration space allocated to primary PCI bus 357 and its subordinates. Optionally, storage 149 may include the memory space or I/O space allocated to primary PCI bus 357 and any subordinates. In this embodiment, pursuant to Plug-and-Play ™ protocol, the address space allocated to primary PCI bus 357 typically encompasses the address space allocated to any optional subordinate busses (not shown) and virtual PCI device 160 resulting in easy and efficient decode by host-to-PCI bridge 340 for identifying host bus cycles targeted to virtual PCI bus 357 and any optional subordinate virtual busses.

09839596.062901

[0031] Host-to-PCI bridge 340 may include bridge configuration registers 347 that are consulted by control circuit 148 to identify host bus cycles targeted to the virtual configuration registers 378 of primary virtual P-P bridge 370 and to route these identified cycles to the Host-to-PCI bridge configuration registers 347. Control circuit 148, in this embodiment, may consult both the bridge configuration registers 347 and storage 145 and storage 149 to determine whether to route host bus cycles to primary PCI bus 150.

[0032] Host bus device 210 may snoop host bus cycles targeted to the configuration registers 378 of primary virtual P-P bridge 370, to learn the bus number assigned by a computer program to primary virtual PCI bus 357 and store this information in storage 115, which holds select information obtained through snooping. Systems having only one host bus device 210 may arbitrarily assign a device number to virtual PCI device 160. Systems with multiple host bus devices 210 may require a mechanism for associating each host bus device 210 with a distinct virtual PCI device 360, such as providing each with a unique device number (i.e. slot number), which may be hardwired, preprogrammed, or stored by an initialization program.

[0033] Fig. 4 shows a system 400 for explaining an embodiment of the present invention, pursuant to the teachings for system 300 shown in Fig. 3. Host bus device 410 may conform to the description given for host bus device 210 and 310. System 400 implements mechanism one for accessing configuration space. System 400 is further comprised of a secondary virtual P-P bridge 490, which directly interfaces a secondary virtual PCI bus 451. Secondary virtual P-P bridge 490, may appear from the perspective of a computer program running on a processor 130, as residing on primary virtual PCI bus 357 and virtual PCI device 160 may reside logically on secondary virtual PCI bus 451.

[0034] The secondary virtual bus 451 is subordinate to primary virtual PCI bus 357, and, in this embodiment, the address space allocated to secondary virtual bus 451 and virtual PCI device 160 are within the address space allocated to the primary virtual PCI

10025390.061502

bus 357. Host-to-PCI bridge 340 may consult storage 145 to identify and route host bus cycles targeted for primary PCI bus 150 and its subordinates, and consult storage 149 to identify host bus cycles that are targeted to primary virtual bus 357 and its subordinates. Host bus cycles targeted to primary virtual bus 357 and its subordinates are not forwarded to primary PCI bus 450 but are allowed to be intercepted by a host bus device 410. Host-to-PCI bridge 340 may complete (i.e. terminate) intercepted host bus 430 cycles according to host bus protocol.

[0035] Host bus device 410 may include storage 111, which is coupled to the host bus and accessible by processor 130. Storage 111 may be comprised of device configuration registers 218, which are accessed by host bus cycles targeted to the configuration registers 268 of virtual PCI device 160. In addition, storage 111 may be further comprised of bridge configuration registers 417 which are accesses by a host bus cycles targeted to the virtual configuration registers 497 of virtual secondary P-P bridge 490.

[0036] Similar to system 300, both the host bus device 410 and host-to-PCI bridge 340, in this embodiment, are aware of the bus number and device number in which primary virtual P-P bridge 370 resides. This information may be provided by an initialization program, hardwired or preprogrammed. Additionally, host bus device 410 includes monitor circuit 114, which, in this embodiment, may learn the bus numbers assigned to primary virtual PCI bus 357 and its subordinates by snooping host bus cycles targeted to the configuration registers 378 of primary virtual P-P bridge 370. This information acquired through snooping may be stored in storage 115. In a system 400 that consist of a single host bus device 410, the device number of the secondary virtual P-P bridge 490 may be assigned arbitrarily.

10062605.055896980

[0037] In a system with a multiple host bus devices 410, a mechanism is required to associate each host bus device with a distinct secondary virtual P-P bridge 490 such as assigning a unique device number for the associated secondary virtual P-P bridge 490 to each host bus device 410, which may be hardwired, preprogrammed, or assigned by an initialization program. Each host bus device 410 may consult its internal configuration registers 417, after it is written by a computer program, to determine the bus number assigned to the secondary virtual bus 451 that directly behinds its associated secondary virtual P-P bridge and then arbitrarily assign a device number to each one or more virtual PCI devices 460. This information may be consulted by monitor circuit 114 to identify of host bus cycles targeted to the virtual PCI device 160 in a similar manner as described for system 100. Each host bus device 410 may include multiple physical devices which are each associated with a distinct virtual device 160 logically residing on its associated secondary virtual PCI bus 451. Plug and Play ™ programs typically group resources assigned to all virtual devices 160 residing behind each secondary virtual P-P bridge 490 resulting in easy decode of host bus cycles by each host bus device 410 for identifying cycles to be intercepted. For example, a single memory address range may be allocated to the secondary virtual P-P bridge 490 that encompasses the multiple ranges allocated to each device.

09896305.062901

[0038] Fig. 5 shows a flow diagram for explaining a method 500 of the present invention that may be utilized by systems 100, 200, 300, 400 and executed by monitor circuit 114. Start 510 may be a host bus reset, which will result in storage and registers being set to default values. The following steps may be executed for each host bus cycle. The step of capturing 520 involves waiting for a host bus cycle, then receiving and latching select host bus address and control signals that indicate the target for the current host bus cycle. The next step of assessing (step 540) includes evaluating each captured cycle to determine whether to enter intercept step 550 or to enter snooping step (step 560). Snooping (step 560) involves receiving and storing in storage 115 select host bus data signals 120. The step of assessing (step 550) includes evaluating each captured cycle to determine whether to do nothing for the current cycle and enter step 520 to capture the next host bus cycle or to enter intercept the cycle (step 580), which causes the current host bus cycle to be routed to access (i.e. read or write) the appropriate locations within host bus storage 111. Step 540 and 550 may be executed in parallel or either step 540,550 may precede the other.

sub A4

~~[0039] Fig. 6a and 6b show flow diagrams for explaining a method 600 executed by monitor circuit 114 of system 400, which is shown in Fig. 4. Start (step 610) and capturing (step 620) may be the same as described for steps 510 and 520, respectively. Assessing (step 540) and snooping (step 560) are accomplished with steps 641, 642, 662 and 664. Likewise assessing step 550 and intercepting step 580 are accomplished with steps 643, 644, 645, 682, 684, and 686. Assessing steps 641-645 may take place substantially in parallel during the address phase of a host bus cycle, and if snooping step 662, 664 or intercepting step 682, 684, 686 is executed, then the snooping or intercepting step may take place during the data phase of a host bus cycle. Snooped cycles may be completed (e.g. terminated) by a device other than host bus device 410 and intercepted cycles may be completed by the host bus device 410.~~

10063963.062501

[0040] Assessing step 641 includes an evaluation of whether the current captured host bus cycle is targeted to configuration-address register 243. A positive evaluation results in step 662 being entered and the data transferred by the current captured cycle is snooped and some or all of this data is stored in mirror register 216. A negative evaluation results in step 642 being entered.

[0041] Assessing step 642 includes an evaluation of whether the captured host bus cycle is targeted to the specific virtual configuration registers 378 for assigning a bus number to primary virtual P-P bridge 370. A positive evaluation requires configuration-address register 243 (as reflected by contents of mirror register 216) to have a value indicating that configuration space is enabled and a value that is currently pointing to the specific configuration register 378 of primary virtual P-P bridge 370 that specifies the bus number assigned to primary virtual bus 357. A positive evaluation also requires the captured read or write host bus cycle to be targeted to configuration-data register 244. A positive evaluation in step 642 results in snooping step 664 being entered where some or all host bus 120 data transferred in the current cycle being latched and stored in storage 115.

09896395.062901

[0042] Assessing step 643 includes an evaluation of whether the current cycle is targeted to configuration space of a secondary P-P bridge 490. A positive evaluation requires configuration-address register 243 (as reflected by contents of mirror register 216) to have a value indicating that configuration space is enabled and has a value that is currently pointing to the configuration registers 497 of secondary P-P bridge 490. A positive evaluation also requires the captured cycle to be a read or write host bus cycle targeted to the configuration-data register 244. A positive evaluation in step 643 results in intercepting step 682 being entered to route the current host bus cycle to configuration registers 417 which are for secondary virtual P-P bridge 490, wherein the specific configuration registers 417 accessed are determined by the current contents of mirror register 216.

[0043] Assessing step 644 includes an evaluation of whether the captured host bus cycle is targeted to configuration space of a virtual PCI device 160. A positive evaluation requires configuration-address register 243 (as reflected by contents of mirror register 216) to have a value indicating that configuration space is enabled and a value that is currently pointing to the configuration registers 268 of virtual PCI device 160. A positive evaluation also requires the captured cycle to be a read or write host bus cycle targeted to the configuration-data register 244. A positive evaluation in step 644 results in intercepting step 684 being entered to route the current host bus cycle to access the particular host bus configuration register 218 as indicated by the contents of mirror register 216.

[0044] Assessing step 645 includes the evaluation of whether the captured host bus cycle is targeted to memory or I/O address space allocated by a computer program to virtual PCI device 160. If the specific configuration registers 218 that specify memory or I/O space allocated to virtual PCI device 460 have not been previously set, then the results of this evaluation is always negative. The address space allocated to virtual PCI device 160, as determined by the current contents of PCI configuration registers 218, are compared to the captured host bus cycle information. If the target for current host bus cycle is within the address space allocated to virtual PCI device 160, then step 686 is entered to route the current host bus 420 cycle to the appropriate internal storage 111 as indicated by the address and control signals of the captured host bus cycle.

[0045] One skilled in the art will appreciate that functions described herein may be implemented in other physical devices than described while keeping with the sprit of the invention.

09825296.010502

[0046] Although the present invention is described as being applied to PCI systems, one skilled in the art will appreciate that the methods taught herein may be utilized by any system having a host bus and a peripheral bus (similar to a PCI compliant bus) where advantage is obtained by having a device on coupled to the host bus appear, to computer programs running on a processor in the system, as a residing on a peripheral bus.

09895395.062901